Interactive Multi–Dataset Visualization and Assimilation Challenges

Todd Plessel

Lockheed Martin
plessel.todd@epa.gov

May 2005

What are the current capabilities and approach for Interactive Multi-Dataset Visualization?

+ Examples
+ Basic process steps
+ Challenges & advice for improving the process

How is Assimilation more difficult?

Future needs (desires) of "Grand Initiative Projects"

Challenges & advice for implementing such projects

Interactive

+ Data / image updates *several times per second*
+ User can pick, zoom, rotate viewpoint *instantly*

Multi–Dataset

+ Bring together data of different kinds/sources
+ Variety of files and formats

Visualization

+ Using computer graphics to depict the unseen
+ Leverage visual part of brain to gain insight
+ Need not appear photo–realistic

Assimilation

+ Compute with the data: cross–media model

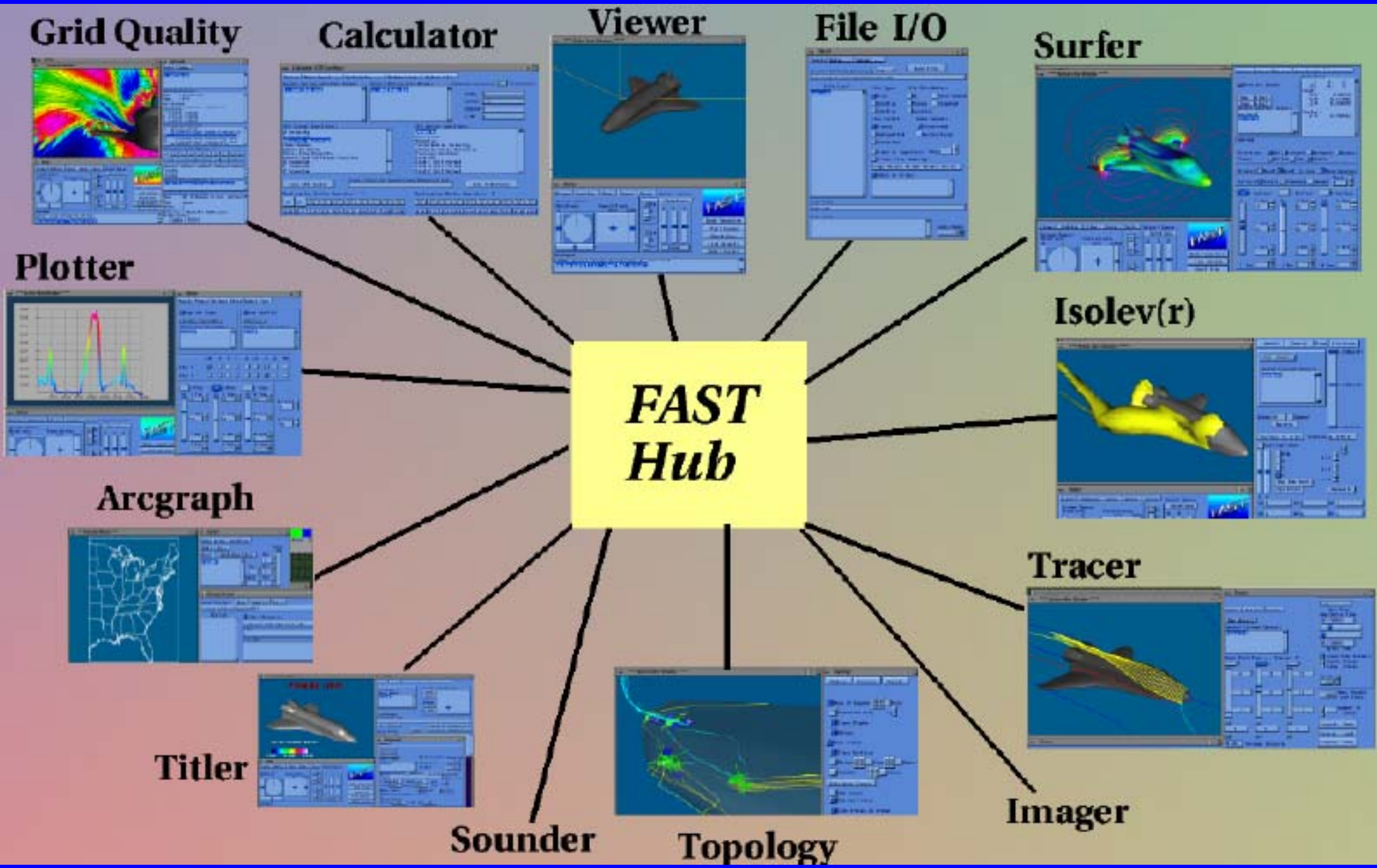
Challenges

+ What is difficult about this?

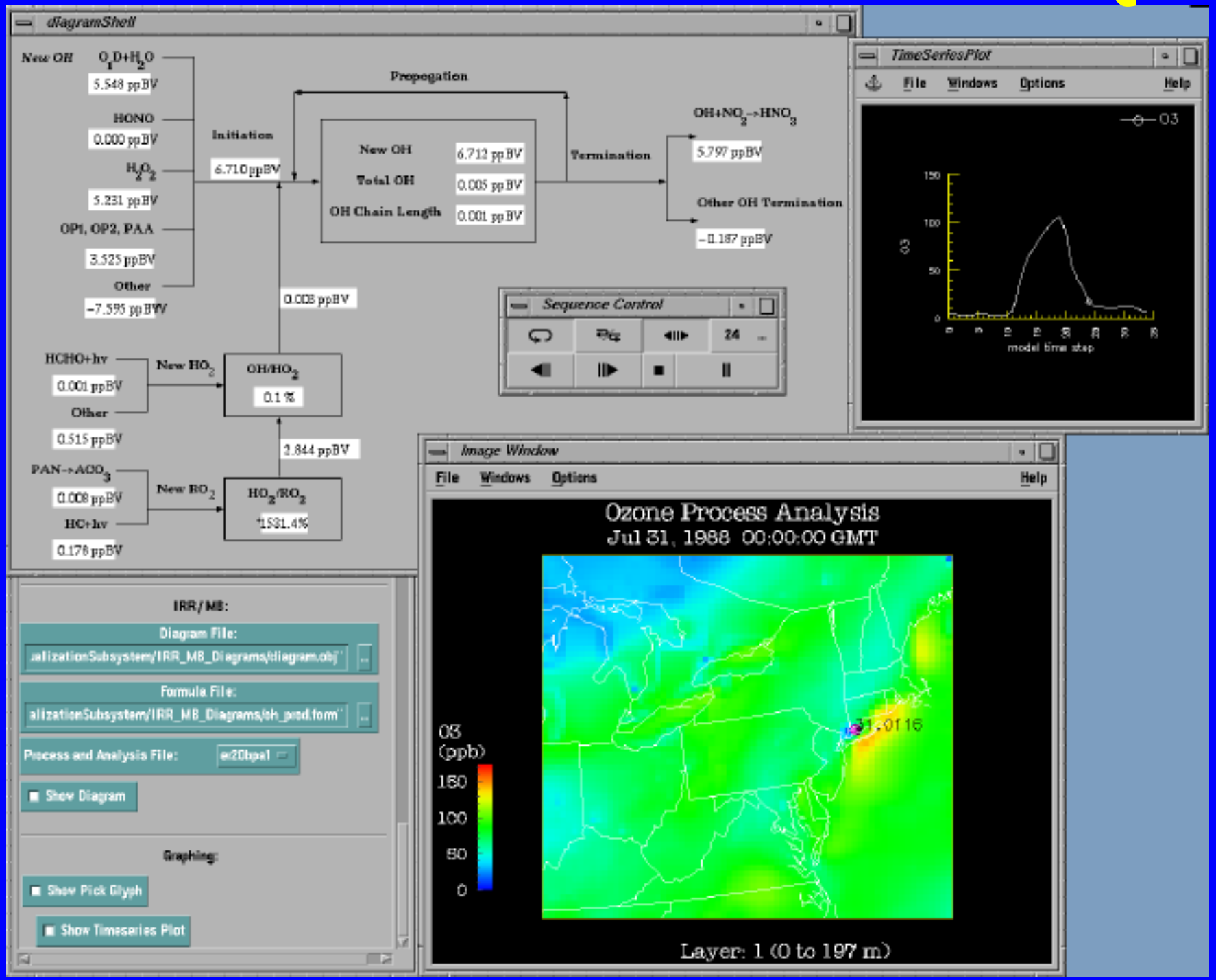
Examples

(screen snapshots) of
interactive multi–dataset visualizations

FAST: Flow Analysis Software Toolkit

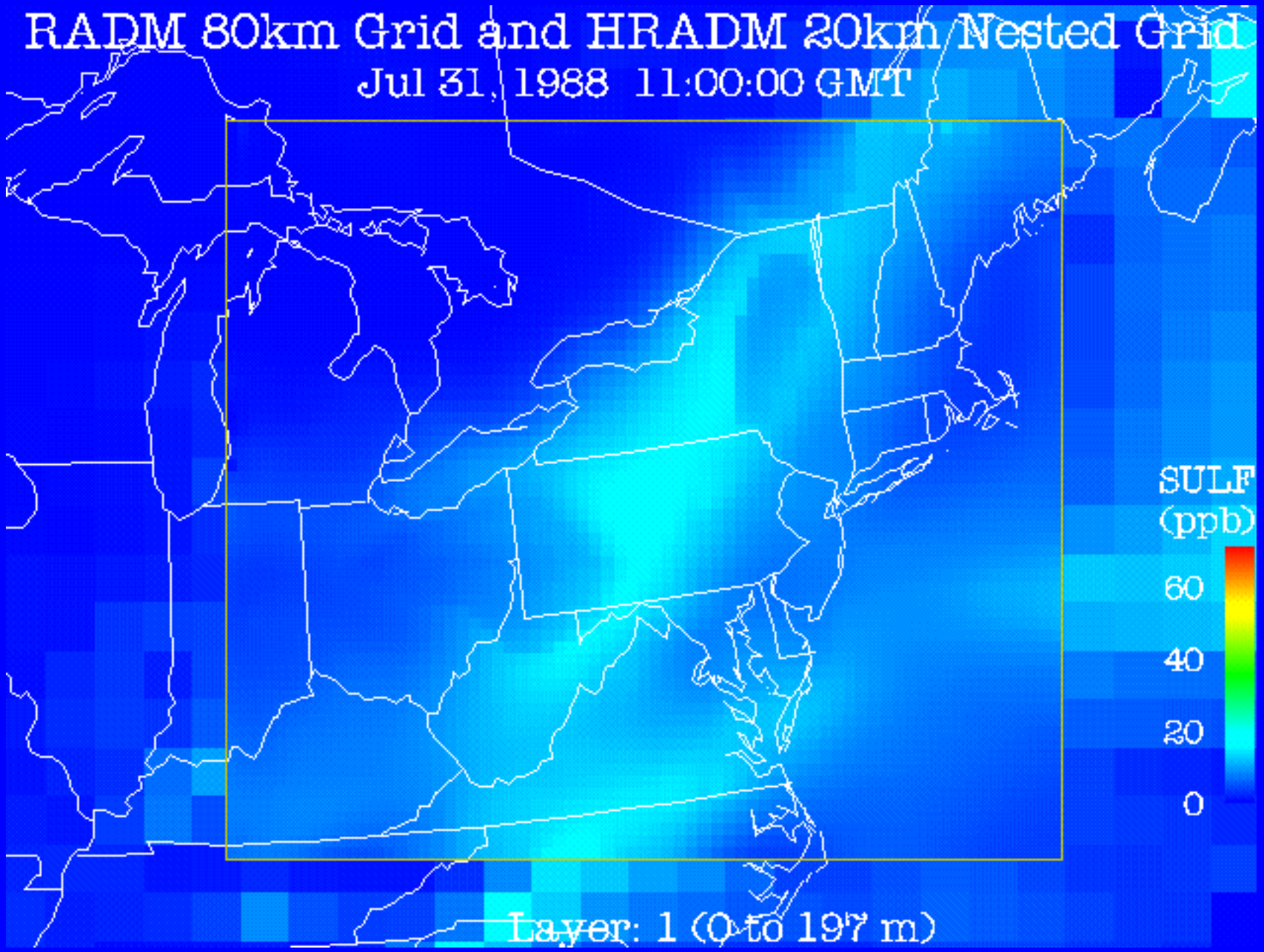


DXDriver: Visualize Models-3 AQ data



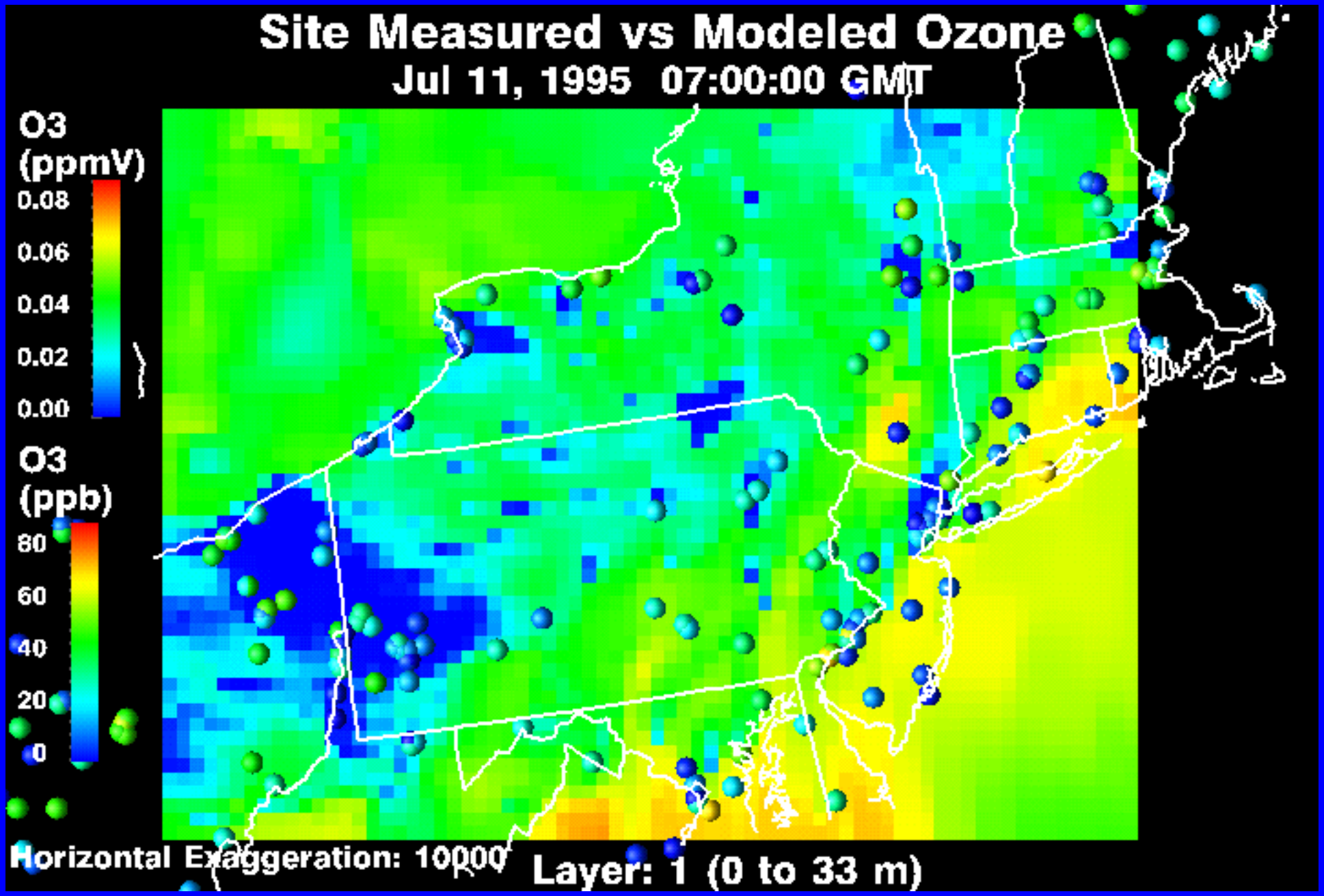
Multiple/Overlapping Gridded Datasets

Map lines, Lambert-Projected, Common Units (ppb)



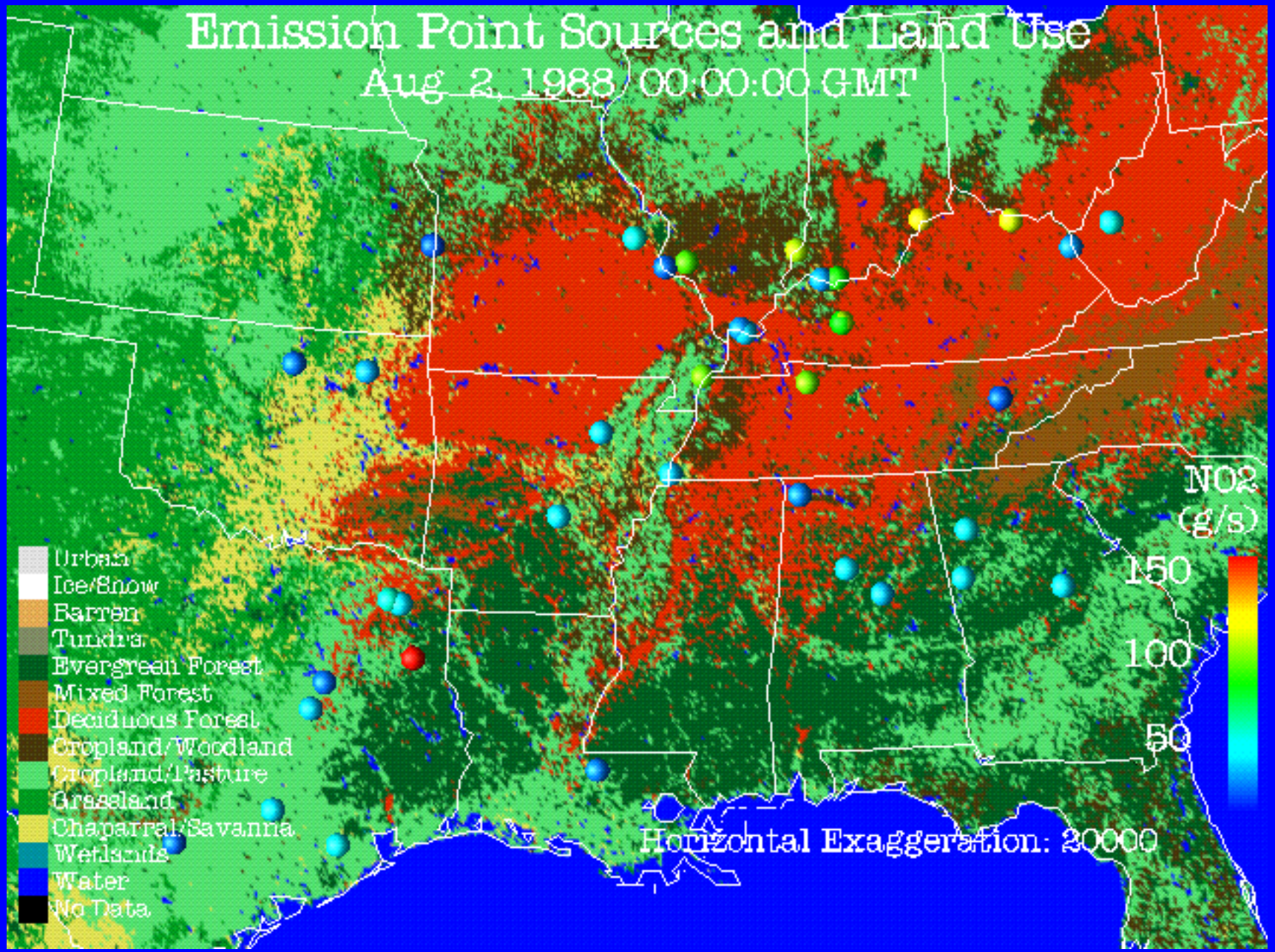
Add Surface Site Observations

Different Units, Project site lon-lat to Lambert

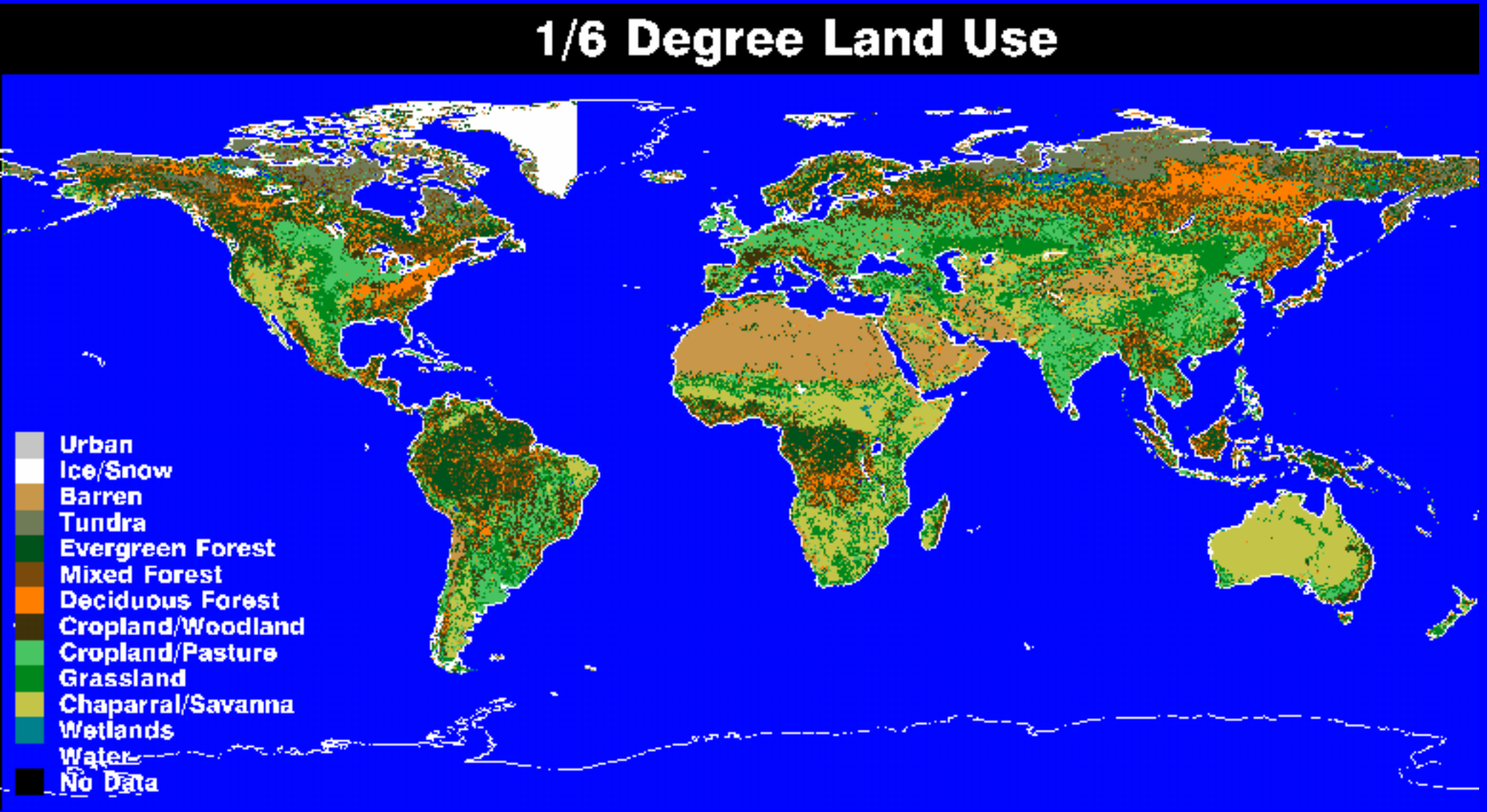


Add Land Use Classification

Project land-use cells from lon-lat to Lambert

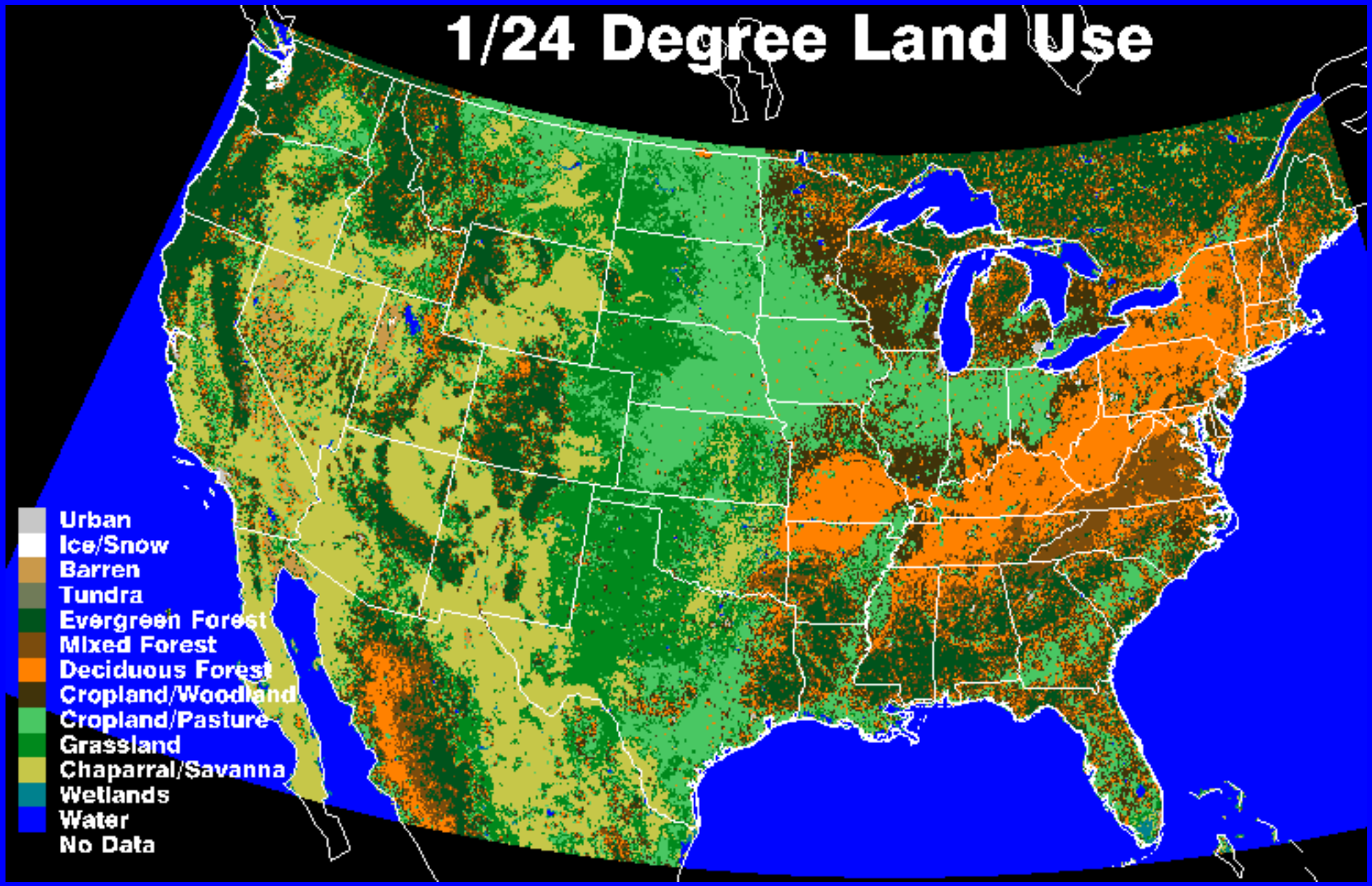


Different Resolution of Datasets



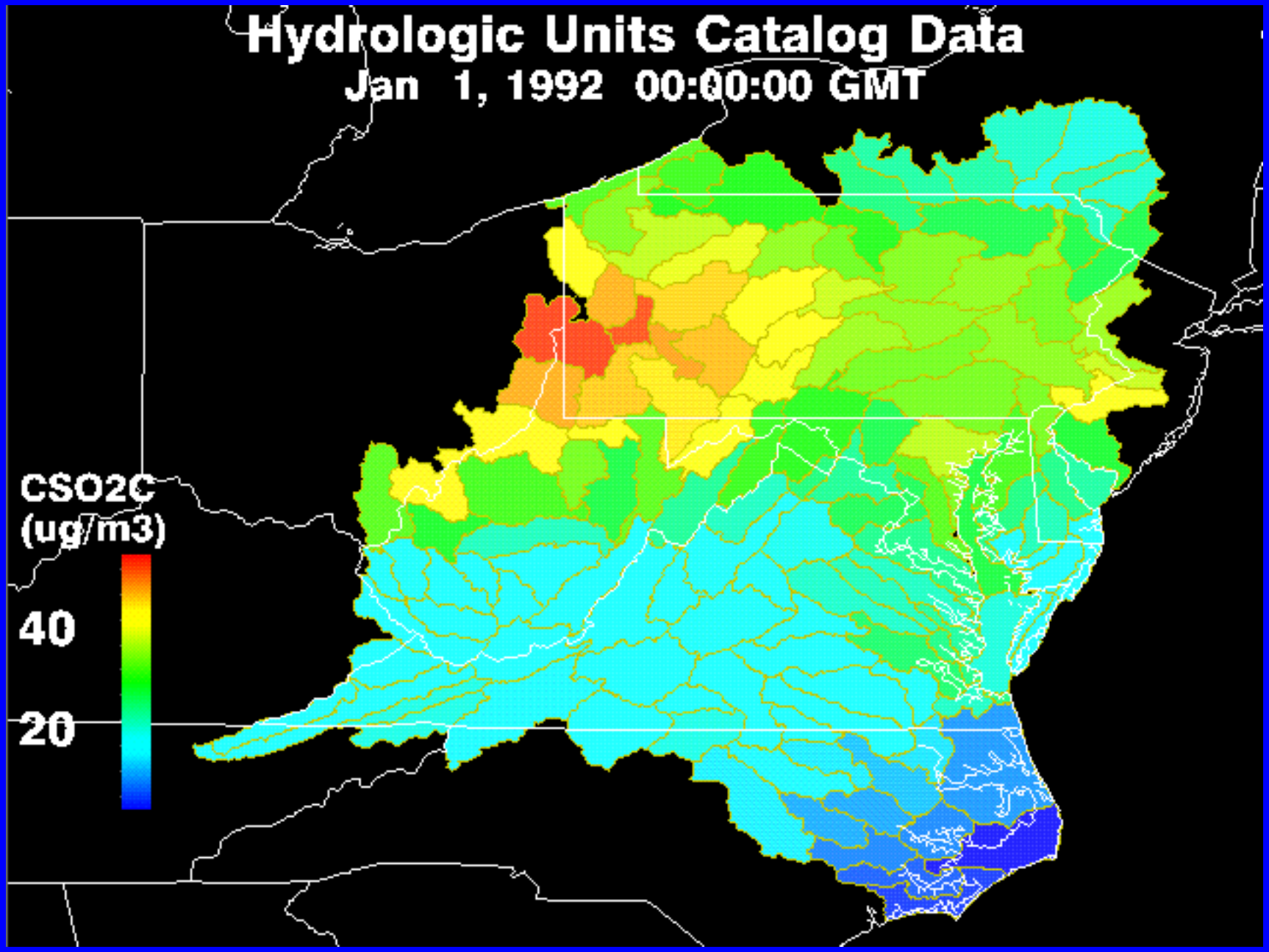
Different Resolution of Datasets

+ Select higher-resolution when available in ROI



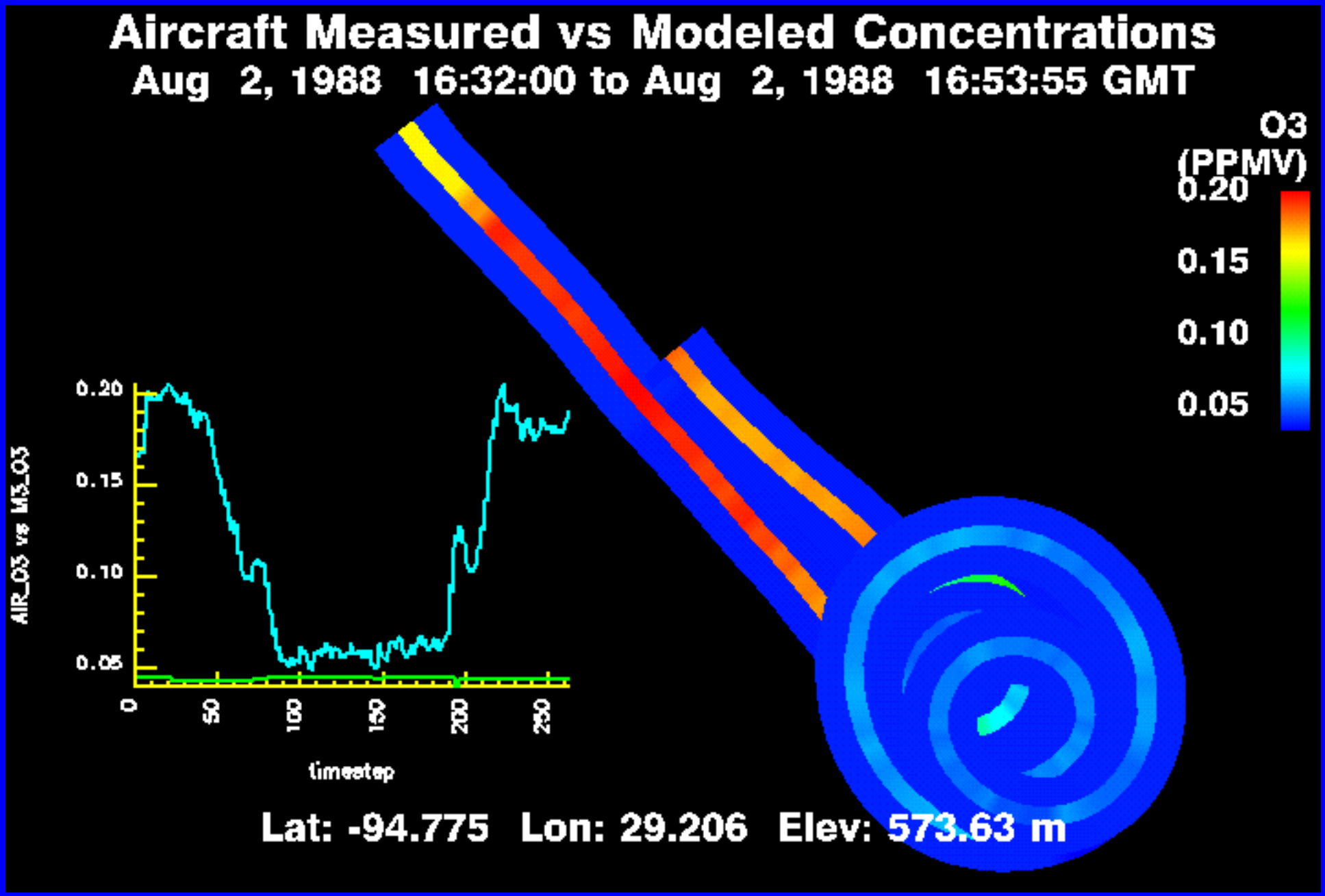
Add Area-Based Data

+ County Emissions (FIPS), HUCS, etc.



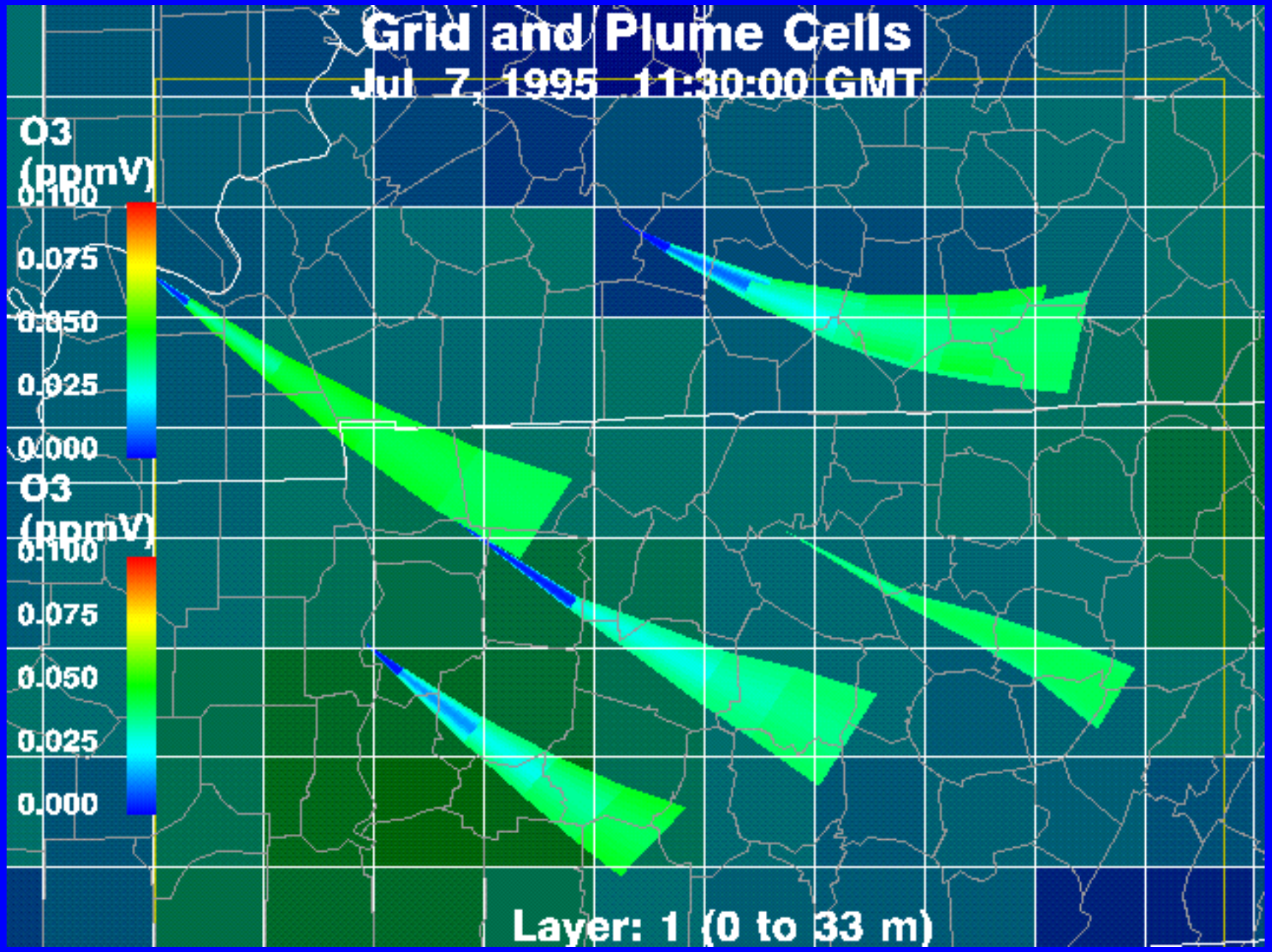
Add Aircraft Measurements

+ Projection, 4D-Interp, Irregular Sample Timestamps



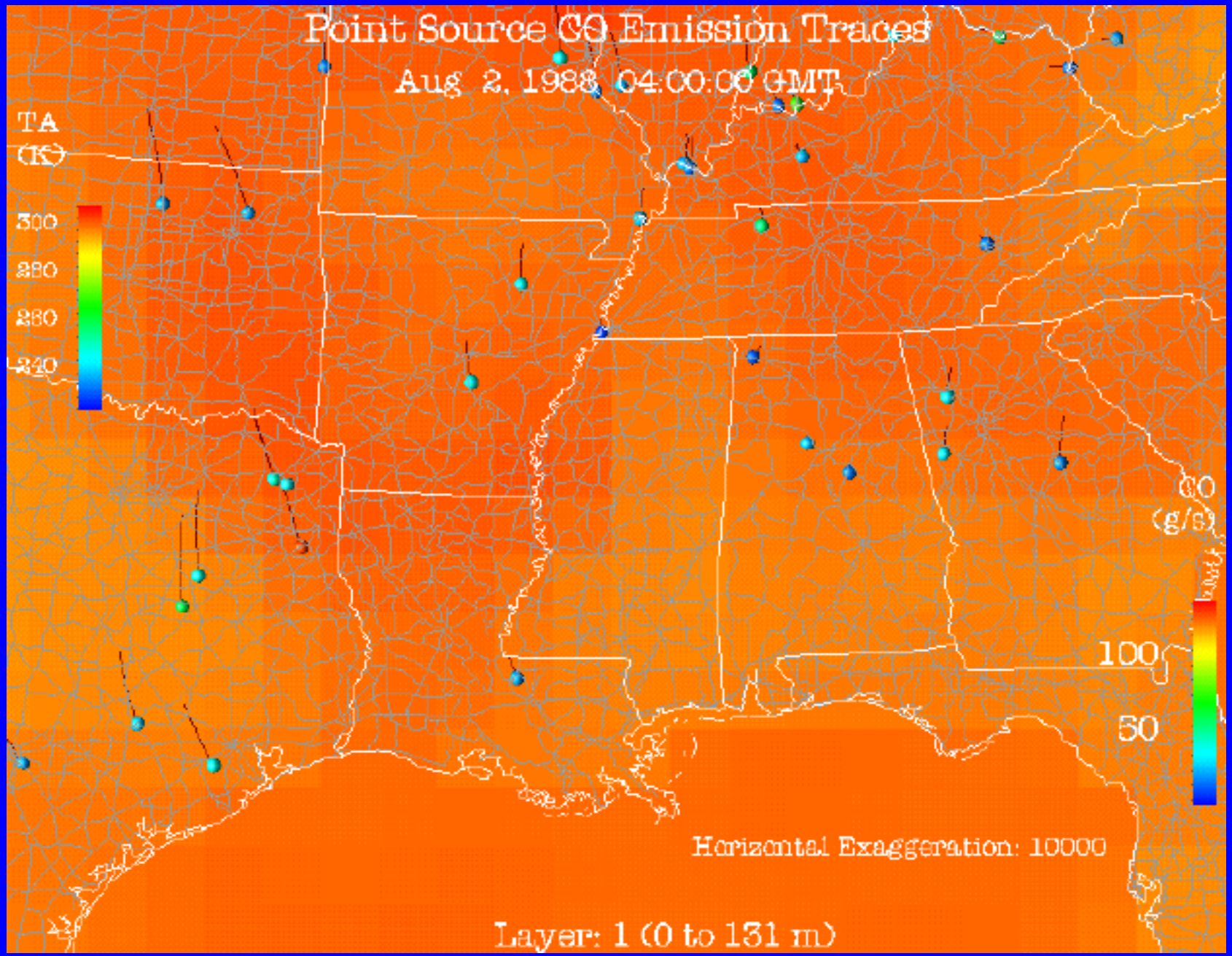
Add Plume Cells

+ Very Skewed Hexahedral vs Tetrahedral cells



Add Emissions Smoke Stacks + Wind

+ Advection of (massless) particle ~= pollution path?



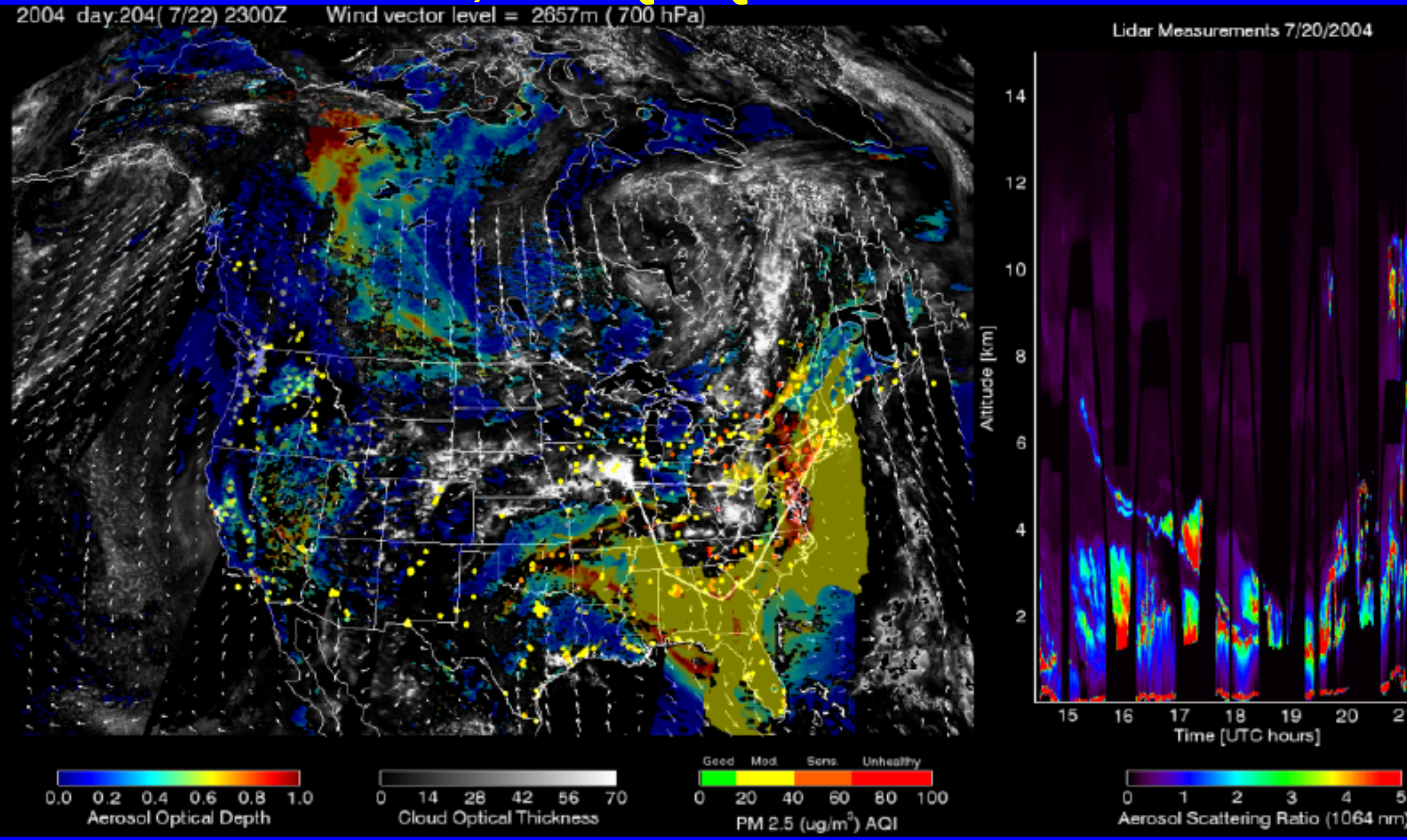
Turn-key vs "Throw-away" one-shot:

+ FAST and DXDriver are custom turn-key vis apps
+ Multi-staff-year development effort
  + DXDriver: 3-staff years, 300KLOC
+ Restricted / a priori set of input files / formats
  + E.g., all GIS data is pre-processed / converted
  + All DXDriver input data files must be M3IO-format
+ Not end-user customizable
+ But can be designed to meet requirements very well

Throw-away vis applications are more common:
+ Meet requirements of a specific short-term project
+ Lower effort to develop (but not trivial)
+ Very limited, hard-coded set of capabilities

Custom "One-Shot Demo" Vis:

+ MODIS satellite, EDAS Wind, AIRNOW Sites,
+ Aircraft LIDAR, CMAQ AQ



Custom "One-Shot" Vis *Process:*

+ Requirements Discussion With Stakeholders
  + Identify Outputs and Constraints
  + Storyboard resulting visualization (e.g., animation)
+ Locate and copy all input data files (HUGE)
  + FTP, Permissions, Disk space
+ Decipher file form and content (data sleuthing)
  + ASCII vs Binary, Endian, FORTRAN record byte counts,...
  + UNITS, Missing values, Range (for color legends)
  + Defects in data
  + Projections
+ Write converter programs (quick-and-dirty)
  + Handle myriad of issues, foremost SUBSETTING
  + Target format of vis tool
+ Develop vis code (quick-and-dirty)
  + E.g., IDL, DX, etc.
+ Render frames of animation (nice-and-slick)
  + Composite various sequences into one movie
+ Deliver first draft results and iterate

Example: MODIS demo vis

+ Over *2 staff–months* of effort
+ Over *60GB* of disk space to hold COPY of data
+ *Many* phone calls and email messages to clarify
+ Over *6,000 LOC* for converters and visualization
+ Result: Success

<plug>
Email me at plessel.todd@epa.gov
We're ready happy and able to repeat this process for *your* data visualization needs!
</plug>

How could this process be improved?

+ Organize, cleanse and subset data at the source

+ Identify only what is needed (for the specific deliverable)
+ Check and fix defective data
  - Smoke stack "exit flow velocity" = 1600 m/s!
+ Subset data to only what is needed (e.g., only selected variables)
  - 2 orders−of−magnitude reduction in file size!
+ Provide all that is needed
  E.g., coordinates that match data values,
  Terrain height (meters above mean sea level)

+ Get serious about Metadata!

+ UNITS, e.g., SI ("go metric every inch of the way!")
  - Temperature = 35 (gee I guess it is really hot or else really cold)
  - Not serious:

```
:VAR−LIST = " ... A25J ... BALD ..."
BALD:long_name = "BALD          " ;
BALD:var_desc = "Variable BALD"
A25J:long_name = "A25J          ";
A25J:var_desc = "hourly wet deposition values"
```

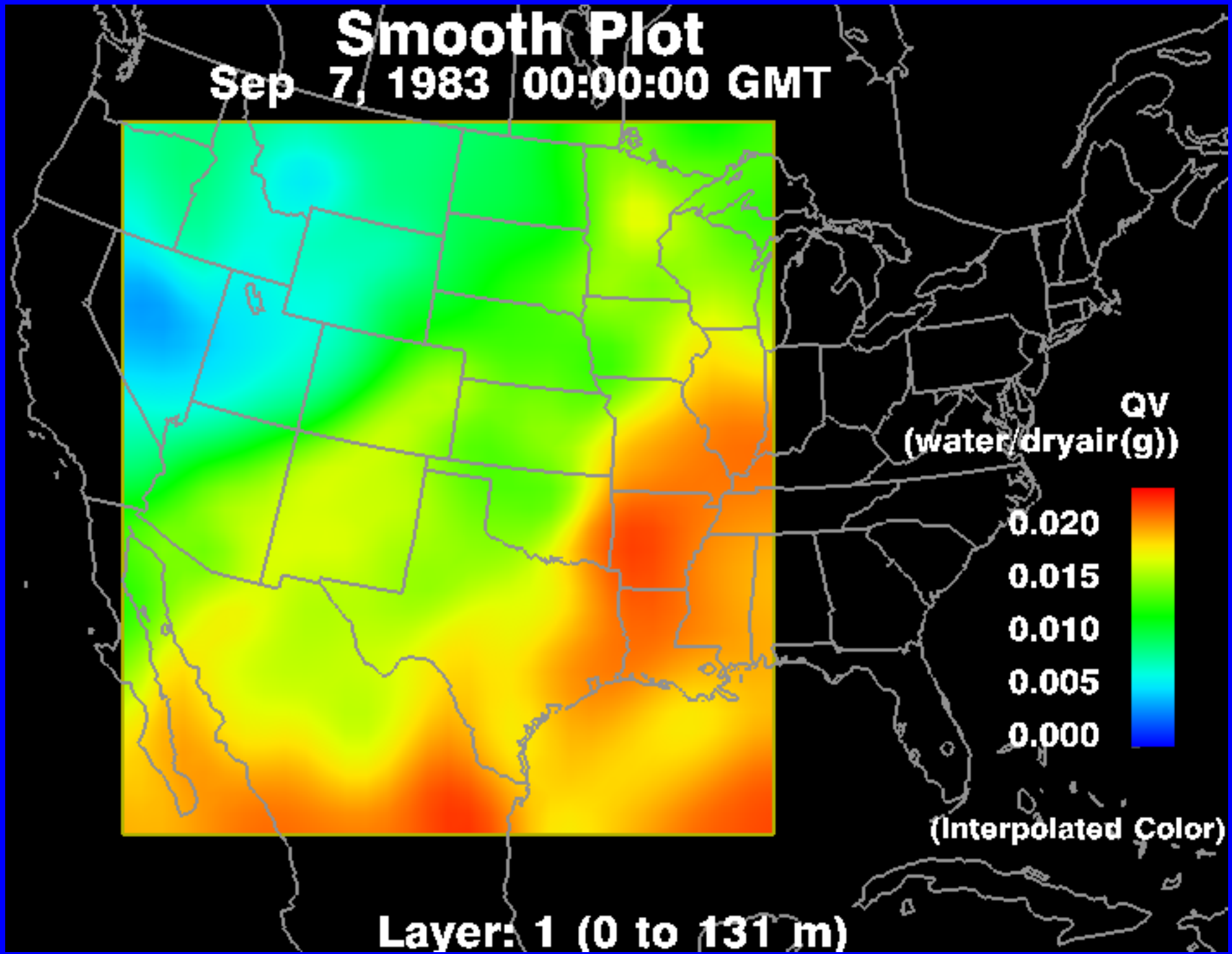
+ Timestamp (e.g., UTC)
+ Accurately Describe Spatial and Temporal Sampling
+ Projection parameters − including Earth ellipsoid major/minor semiaxes

Assimilation is harder than visualization

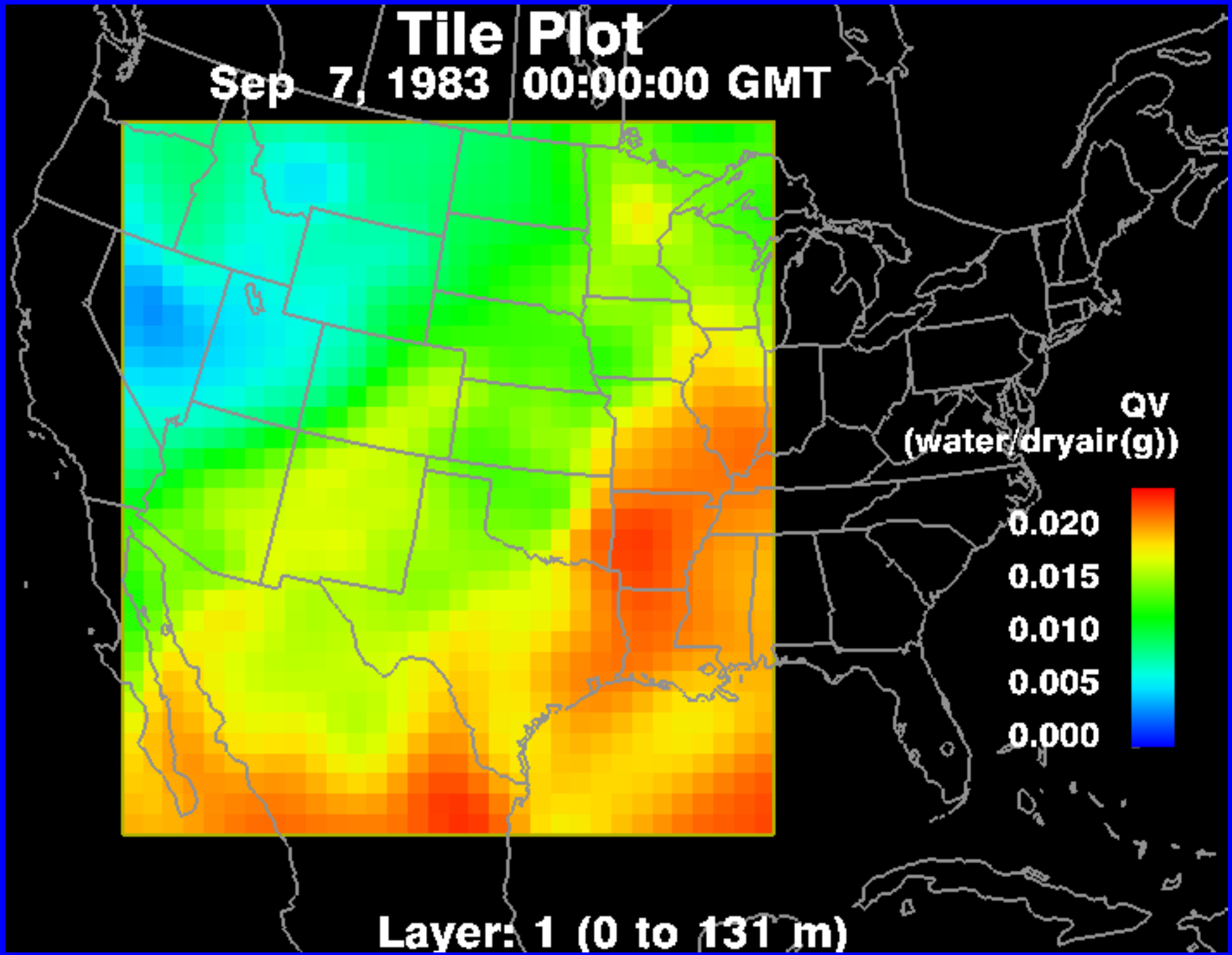
+ Temporal Sampling (instantaneous or averaged):
  + Vis: just display the most recently available data
    E.g., MODIS sat data may be 12 hours old before it is "repainted",
    wind updates only every 3 hours, sites have irregular timestamps,
    LIDAR/TOMS sweep, aircraft opens cannister for 5 seconds...
  + Assimilation: linear interpolation or resampling

+ Spatial Aggregation:
  + Vis: just contour it, show sparse vectors,
    inverse-project then re-project cell centers
  + Assimilation: must handle "polygon intersection"
    problem explicitly or use Krieging,
    finite-difference vs finite-volume methods
    implicit vs explicit ...

Vertex-based: data at cell corners

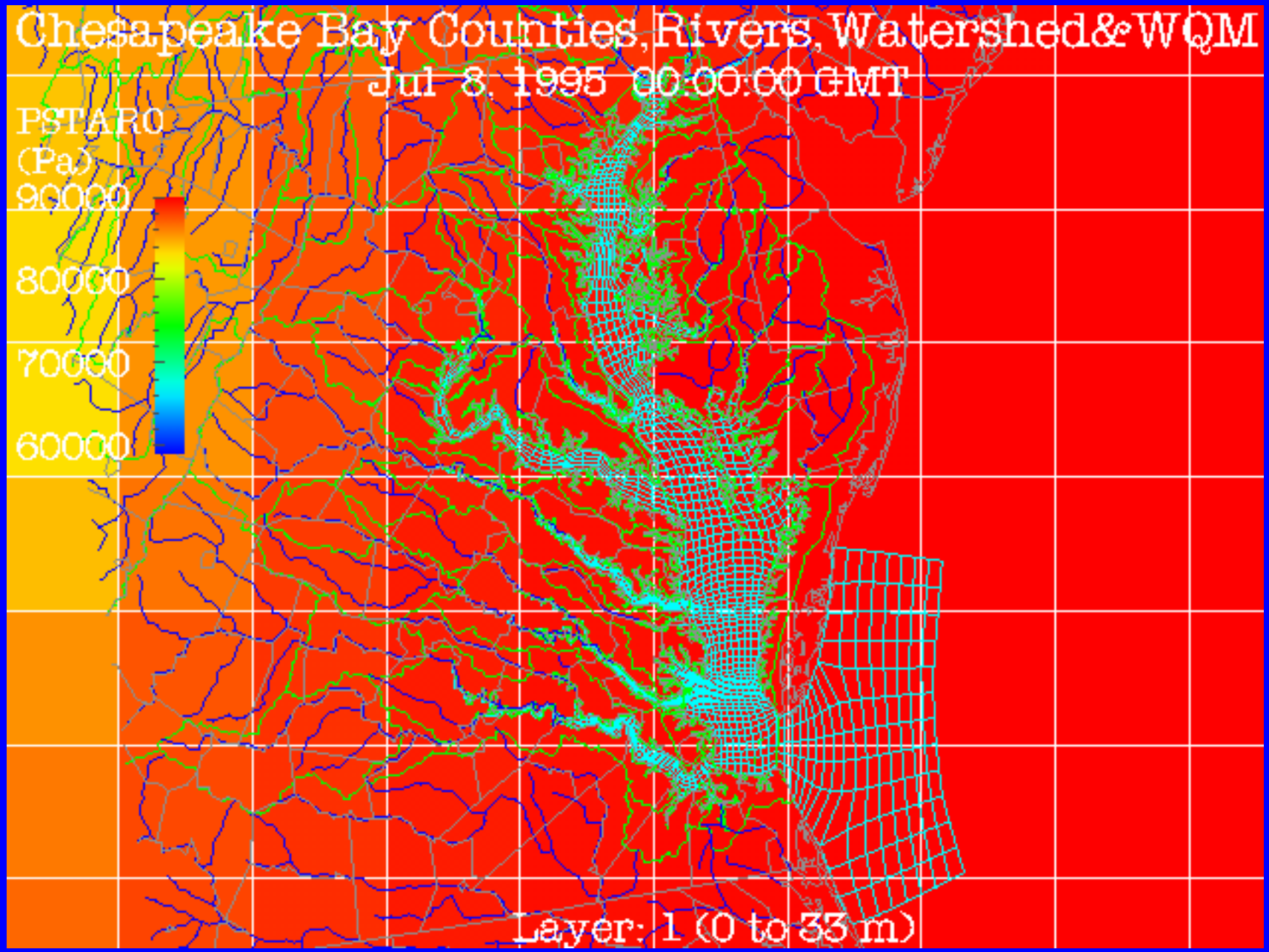


Cell-based: data for entire cell

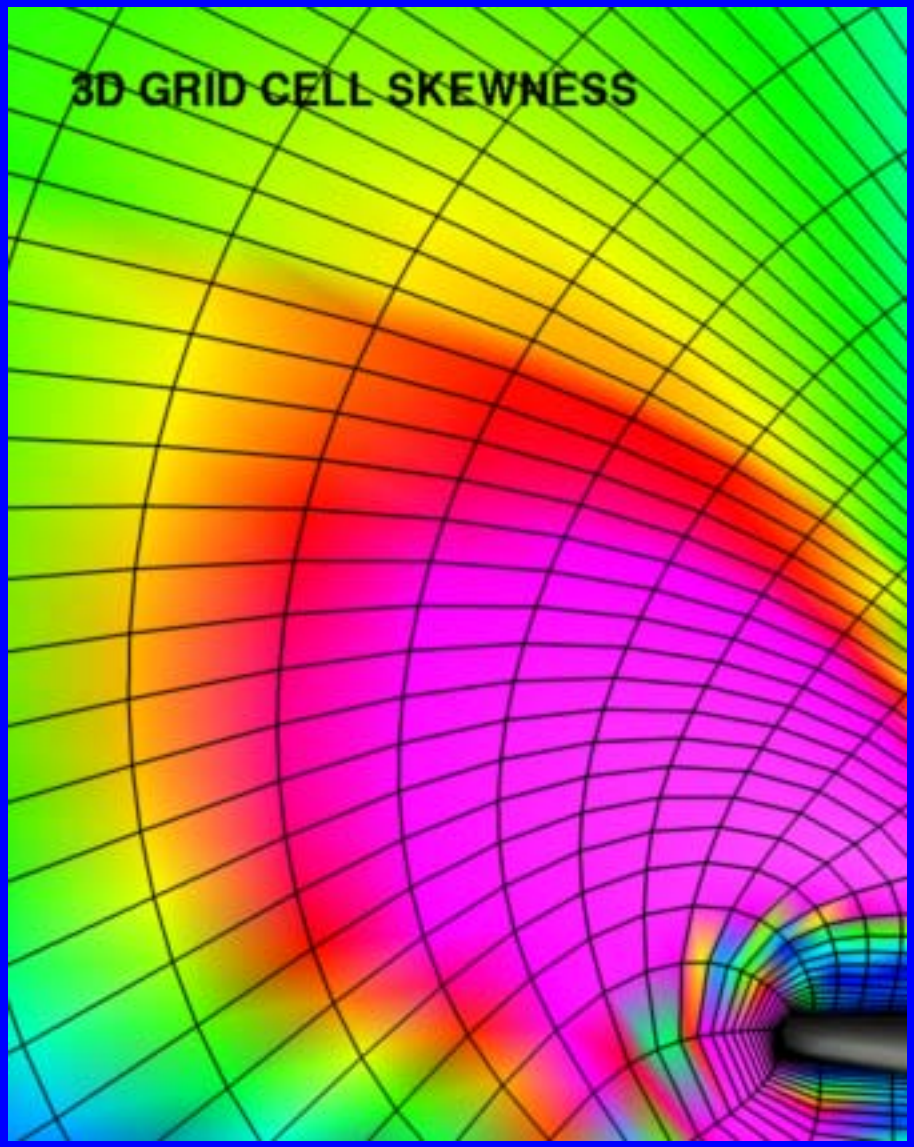


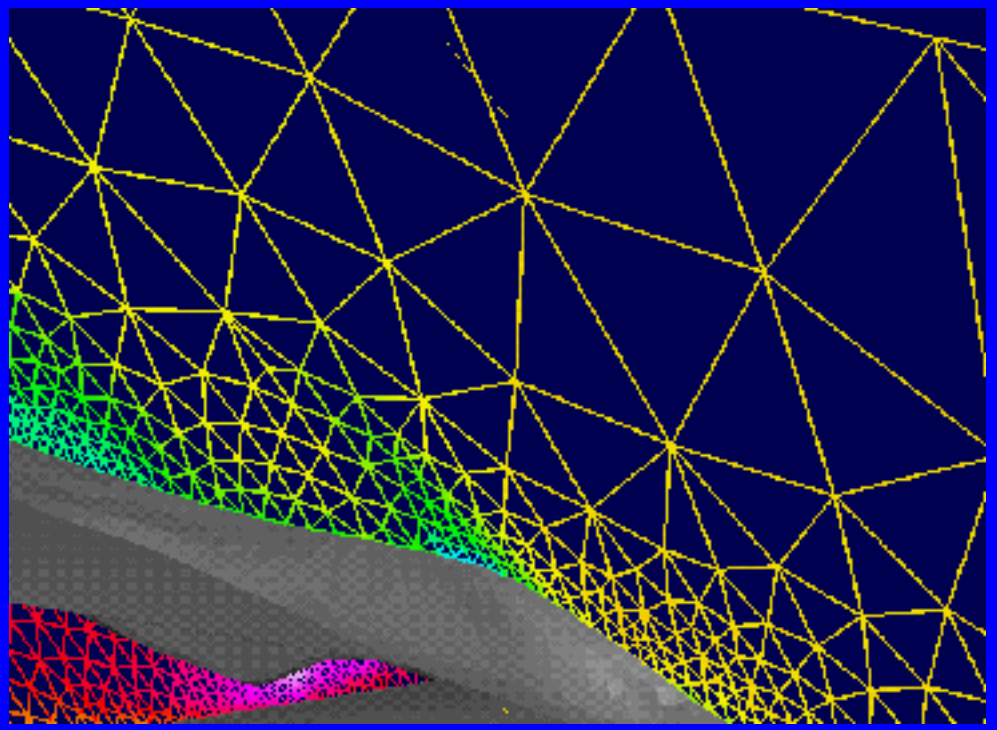
Curvilinear Grids

(this one is *crazy*...)



Issues: Curvilinear Grid Quality vs Unstructured Grid





Challenge: Grid Cell Polygon Intersection

Sample vs Weighted Average vs Mode vs Distribution ...

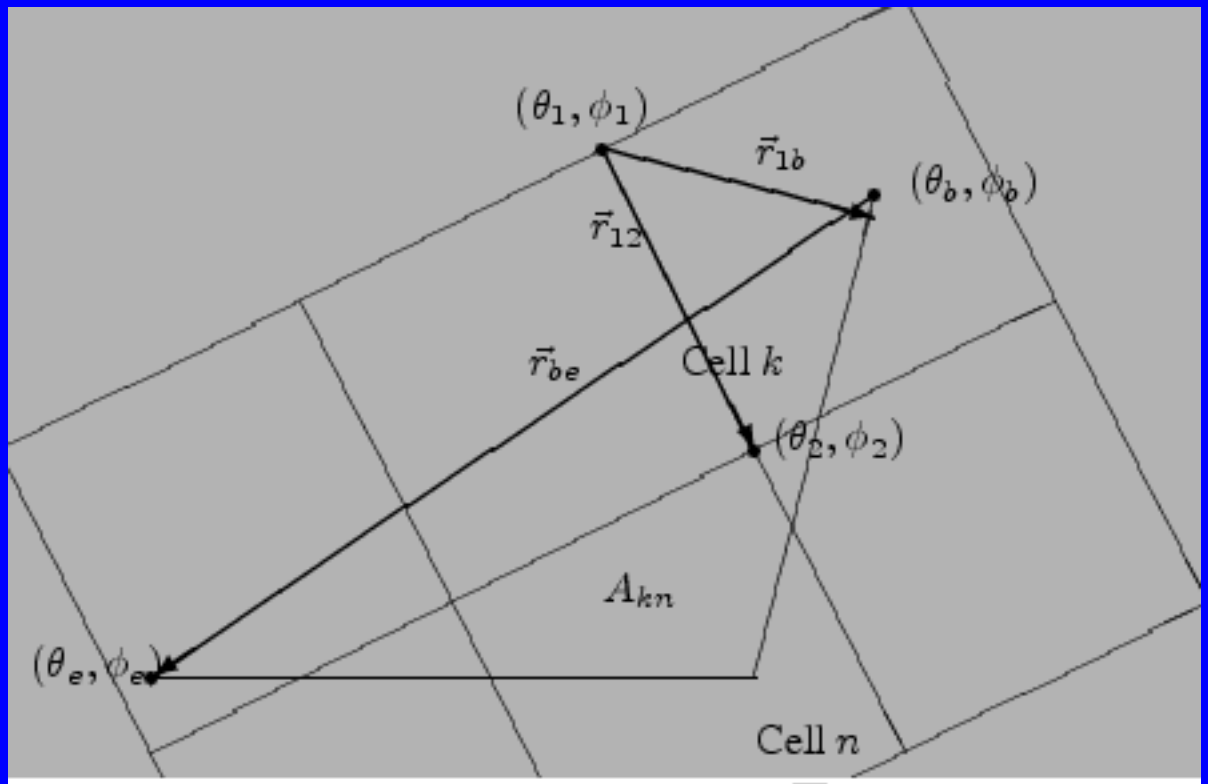


Figure 9: An example of a triangular destination grid cell k overlapping a quadrilateral source grid. The region A_{kn} is where cell k overlaps the quadrilateral cell n. Vectors used by search and intersection routines are also labelled.

(From ESFM: www.esmf.ucar.edu)

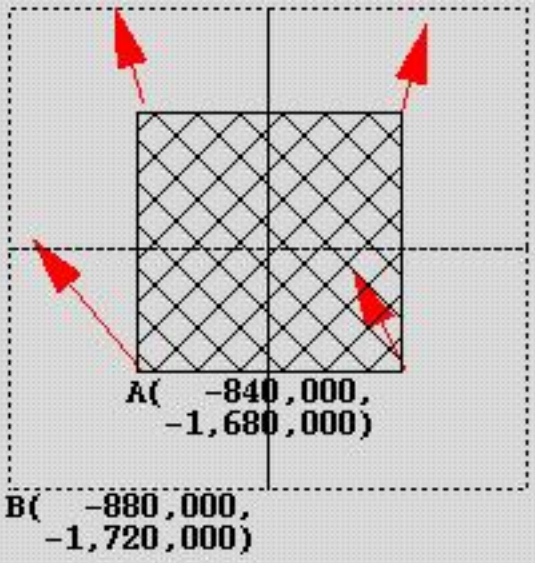
Challenge: DOT vs CROSS / Arakawa Schemes

```
Cross:  NROWS = 1, NCOLS = 1,
        XORIG = -840,000, YORIG = -1,680,000
        XCELL =   80,000, YCELL =     80,000

Dot:    NROWS = 2, NCOLS = 2,
        XORIG = -880,000, YORIG = -1,720,000,
        XCELL =   80,000, YCELL =     80,000
```

The cross-hatched square represents the single cross-grid cell colored by some variable, say, O3. Note that the lower-left corner of the sqaure is at the cross-grid origin (-840,000, -1,680,000).

The four arrows are 2D wind vectors from the dot-grid. They are rooted at the centers of the cells of the dot-grid (dotted line).
For example, the first arrow is rooted at:
dot-grid(xorig + xcell / 2, yorig + ycell / 2) =
(-880,000 + 80,000 / 2, -1,720,000 + 80,000 / 2) = (-840,000, -1,680,000).



Recap: Today's State of the Practice

+ Visualizations are either:
  + Turn-key Apps (large-scale developments) or
  + One-shot throw-aways (immediate-term project)
+ Both are programmer-intensive efforts
  + Detective work, hacking converters, processing, *not amenable to automation*

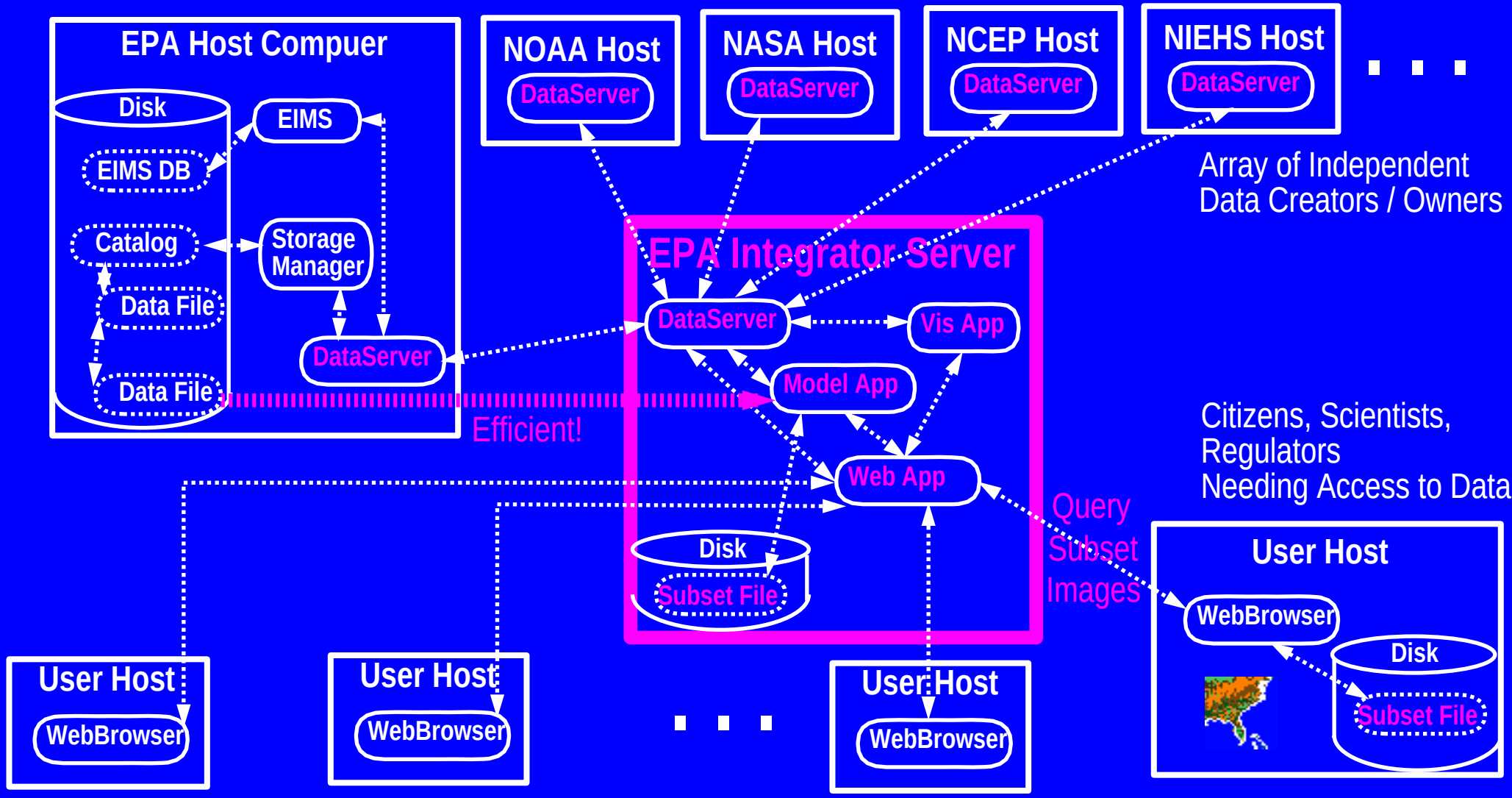
+ Assimilation is in a similar state (but harder)
  + Cross-media models are not very common
  + Input data (met, emis., obs) is carefully gathered (replicated), pre-processed (e.g., regridded), etc.
  + *Typical models don't discover and assimilate arbitrary input data on-the-fly!*

Future Needs of Grand Initiative Projects

+ Automagic discovery of relevant datasets

+ Convenient, efficient, on-demand streaming subsets

+ Sophisticated dynamic adaptive robust models



Challenges of Grand Initiative Projects

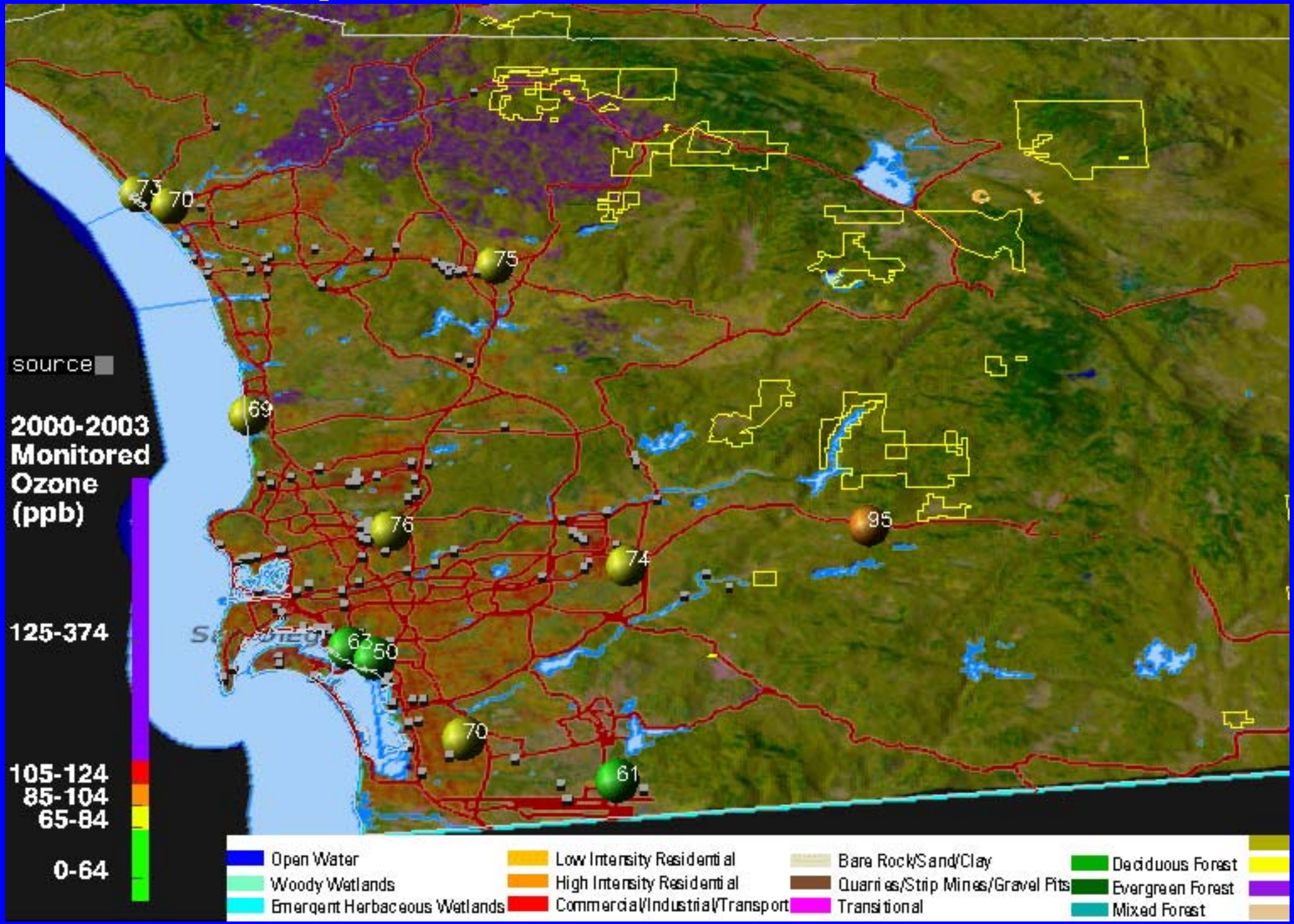
+ Automagic discovery of relevant datasets

+ Web Agents
  + Who has expertise in this new field?
    (Hint: they all work at *Google*)

+ On–line Data Catalogs
  + How do we explain and convince the Data Creators / Owners that they must catalog their available data files (well).
  + E.g., *EIMS (Environmental Information Management System)*

+ Web Servers
  + How do we get them to create convenient Web Server (gateway) apps?
  + E.g., GIS database *canyon:esri_sde* accessible through ESRI SDE API and *emfinder* used by *WME (Window on My Environment)*

Exemplar: WME 3D Visualization



Challenges: Wallclock Time Efficiency

Minimize wallclock time to complete operation

+ Serial vs Parallel (SMP) vs Distributed (LAN vs WAN/Grid) Execution

+ Distribution => additional I/O operations (slow)

+ *Efficient* compared to what?

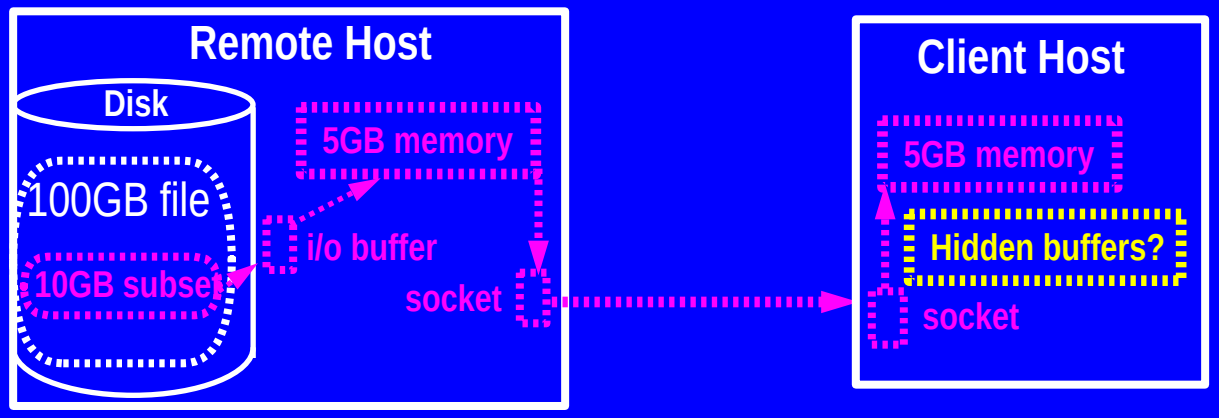
+ Non-distributed execution:
  1. Local binary file `fseek()/fread()/fwrite()` - FASTEST!
  2. NFS-mounted file (about *50 times slower*)
+ Distributed execution (about *1.5 - 10 times slower i/o*):
  3. LAN `socket()`
  4. `MPI_Send/Recv()` (about *20% slower than sockets*)
  5. General-Purpose Distributed Computing APIs (RPC, JavaRMI, ...)
  6. Grid / Middleware (Globus, Legion, OpenGrid, ...)

"Abstraction Penalty":
More layers of code allow a greater chance for hidden inefficiencies

Challenges
Memory Efficiency

Minimize redundant / unneeded / "hidden" data copying

+ Data (subset) need only move from disk to memory (1 copy) and if distributed, also to socket output buffer (repeated) to remote host socket input buffer (piecewise) to memory.
+ So there are 2 copies of the data: disk + local memory + remote memory + 2 x socket buffer size (small ~ 256KB). Anything beyond this is wasteful!
+ Carefully trace through your distributed application (and the muddleware libraries it uses) and you will likely find many instances of additional wasteful copying...



+ Any hidden memory copies would be revealed in large HPC codes:
  E.g., CMAQ Memory (in-core): US domain at 1km grid point spacing =
  2 timesteps x 50 variables x 75 layers x 3240 rows x 4752 columns x 8 bytes/real

= ~ 1TB

Challenges

+ *Convenient, efficient, on-demand streaming subsets*

+ *Convenient* means in a form easiest to compute with (REAL*8 array, object)
  + Powerful Data Model and supporting API (e.g., Object-Oriented Classes)

+ *On-demand* based on user interaction with web-based application
  + E.g., WME

+ *Streaming* means from remote disk file into local host memory (array/object)
  + No file replication (FTP, GridFTP)
  + E.g., ESRI SDE `SE_shape_get_all_points()`
  + Others: OpenDAP? ESMF?

+ *Subset* means only selected variable at/over subspace and timestamp
  + E.g., ground-layer ozone over Charlotte at 8am UTC
  + DXDriver invokes M3Subset and achieves such extracts from a remote host file in 1/10th of a second to allow interactive visualization!

Challenges: Powerful Convenient *Data Models*

What is a Data Model?

A coherent set of mathematical abstractions useful for representing typical scientific datasets and operations on them.

Examples include: *Fiber Bundles* [Butler & Pendley], *Vector Bundles* [Butler & Bryson], the *Lattice* [Hibbard] and *Field* [Collins, et al.].

But mathematical elegance alone won't help us meet out computational modeling goals – we need high–quality *software libraries*.

See my survey of selected data model software:

http://www.cs.unc.edu/~smithja/MIMS/DataModel/research/DataModelReport.html

Example: My *Field Data Model* Work

http://www.cs.unc.edu/~smithja/MIMS/DataModel/development/design/overview.html

A collection of high–quality multi–language software libraries for efficient representation, integrated analysis and visualization of large diverse time–varying geospatial environmental data and metadata for supporting cross–media modeling and decision–support applications operating in a high–performance networked multi–platform computing environment.

It is designed to handle every kind of data I've encountered in my career, *well*.

Field Data Model Native File Format

+ *XML* (eXtensible Markup Language) xml.org
  + For compatibility with external applications
  + Custom DTD: *FDML (Field Data Markup Language)*
+ *XDR* (eXternal Data Representation)
  + IEEE-754 reals and "big-endian integers"
+ ASCII "header/metadata" files with references to binary data files.
+ Allows referencing into existing NetCDF files.
+ Custom "Streamers" for import/export of Field Data Model objects from/to foreign format files (e.g., M3IO, Vis5d, etc.).

How does it work?

Some key elements:

+ *DataObjects* are used by OB/OO (e.g., Fortran-90, Eiffel, Java) Applications

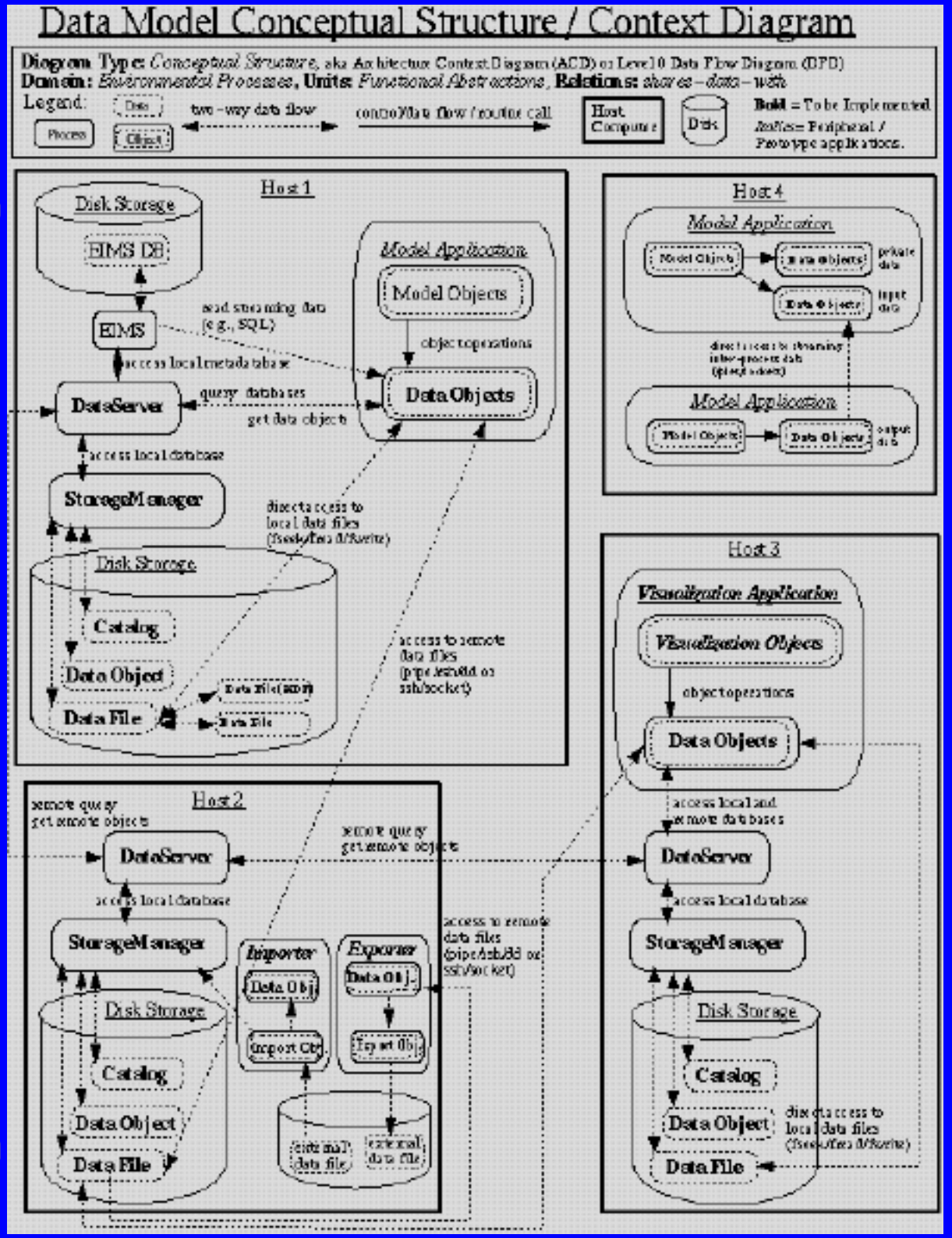
+ Applications can query *DataServer* applications to obtain desired DataObjects.

+ *DataServers* communicate with each other and with *StorageManagers*.

+ *StorageManagers* maintain their own database *Catalogs* of data files. *EIMS* and *ARCinfo* can be examples.

+ Once created, *DataObjects* have direct (O(1)) access to the local or remote data files (via fseek/fread, pipes or sockets).

+ *Streamer* applications can be created to import/export *DataObjects* from/to foreign file formats (e.g., M3IO, Vis5d,...)



Field Data Model Native File Format

Basic "may contain" hierarchy of data classes:

- DATA_SET
  - GROUP
    - MULTI_FIELD
      - MULTI_MESH
        - MESH
      - FIELD
        - DATA
          - PRODUCT_ARRAY
            - NURBS
              - STORED_ARRAY
              - REGULAR_ARRAY
              - CONSTANT_ARRAY

E.g., A DATA_SET (time-series) of a GROUP (various species) of FIELDs ("ozone") containing MESH (regular grid in Lambert space) and DATA (fundamental metadata) represented as a STORED_ARRAY (of reals).

Some *Field Data Model* Details:

Fundamental Metadata

STORED_ARRAY

- DIMENSIONALITY
- DIMENSIONS
- CATEGORY (BYTE, TEXT, INTEGER, REAL, COMPLEX, ...)
- RANK
- SHAPE
- BYTE_FORMAT (ASCII, XDR BITS 8,16,32,64)
- LOCATION (FOLLOWS, BYTE offset, FILE, PIPE, SOCKET)

DATA

- VALUES
- VALIDITY (ALL, COMPARED NE,..., TAGGED, INDEXED)
- QUANTITY, UNITS, RANGE, MAGNITUDE_RANGE
- TIME_STAMP (NONE, CLOCK..., DATE, DATE_INTERVAL DATE_DURATION)
- ACCURACY (SIGDIG, INTERVAL, PERCENT, DISTRIBUTION)
- METADATA_FILE

Some *Field Data Model* Details

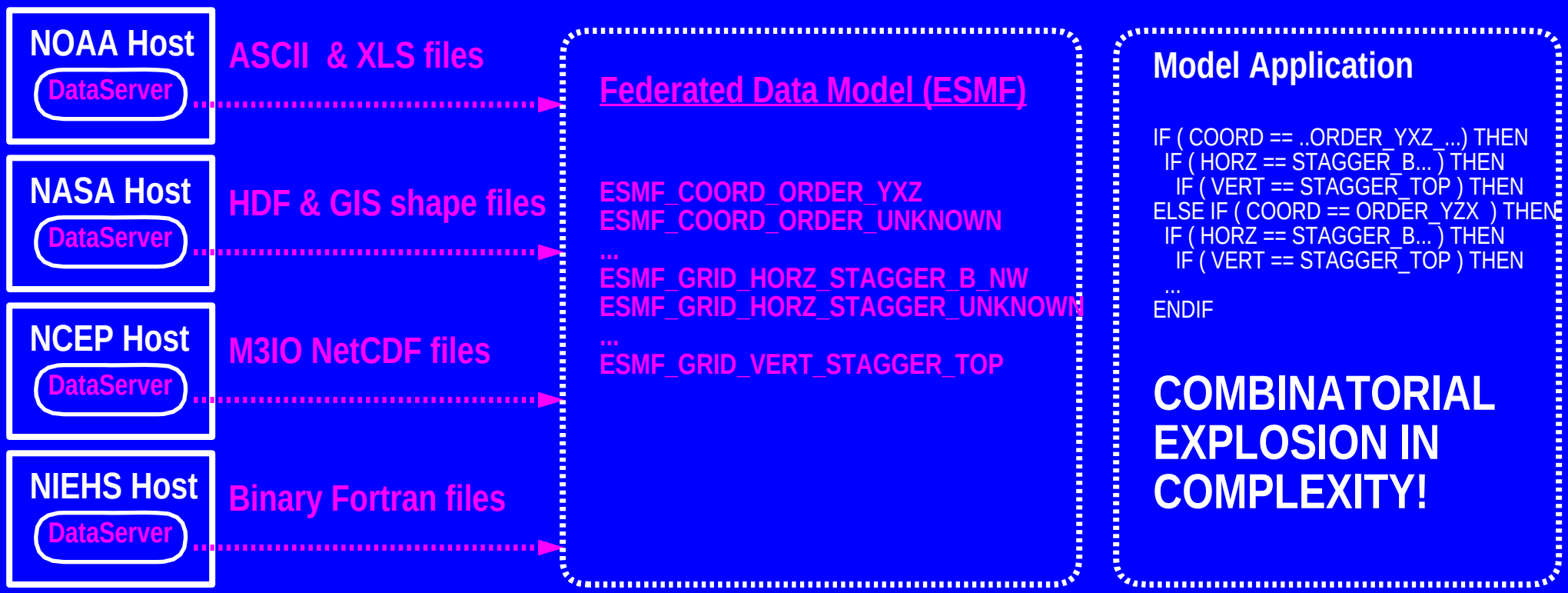
- **MESH**
  - **GEOMETRY**
    - **DIMENSIONALITY**
    - **COORDINATE_SYSTEM (CARTESIAN,... CARTOGRAPHIC)**
    - **VERTICES**
  - **TOPOLOGY**
    - **DIMENSIONALITY**
    - **CELL (POINT,LINE,POLYLINE,...TRIANGLE...TETRAHEDRA)**
    - **CONNECTIONS (NONE,IMPLICIT,INDEXED,COUNTED...)**
    - **VALIDITY (ALL, TAGGED, INDEXED)**
    - **NEIGHBORS (NONE, IMPLICT)**

Data Models: Federated vs Unified

Federated Approach (e.g., ESMF?):

Keep extending Data Model as needed to *describe* every new data form encountered

Array of Independent Data Creators / Owners

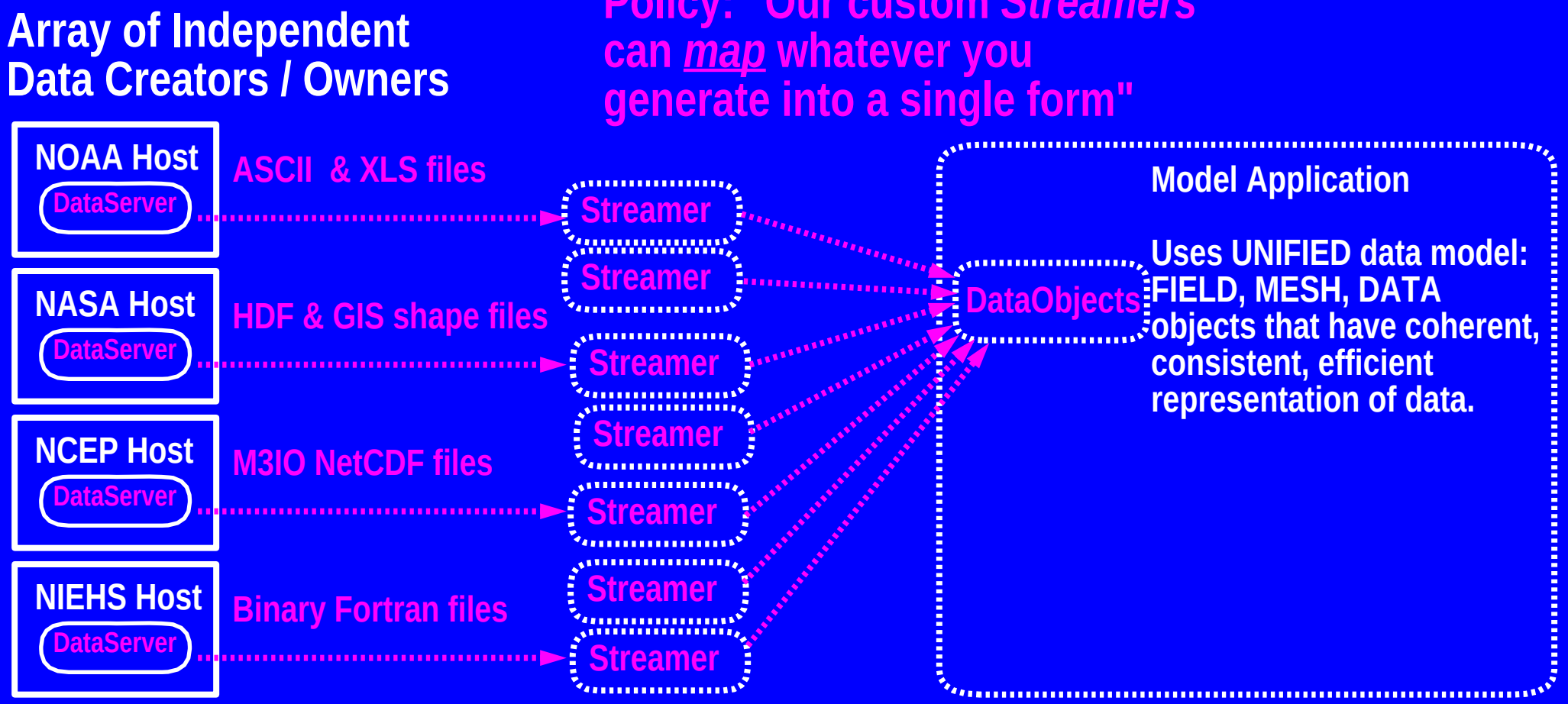


Merely describing the myriad of data just passes the complexity onto applications!

Data Models: Federated vs Unified

Unified Approach (e.g., FDML):

Map all data onto a powerful coherent data model



Mapping the myriad of data forms into a single unified form simplifies applications!

Challenges

+ Sophisticated dynamic adaptive robust models

- + E.g., Cross-media models, human health effects, decision support systems
- + These are very ambitious high-risk projects!
  - + Requires cooperation of and improvements to data creation / owners
  - + Requires powerful, preferably unified, Data Model with high-quality APIs
  - + Requires development of Streamers to extract subsets and map data to DM
  - + Requires (multi) organizational cooperation from top to bottom ("leaves")
  - + Requires staff expertise in high-quality software engineering
  - + Requires stable long-term staffing and funding...
- + How much dynamicism is desirable anyway?
  - + Need to understand, control, verify and reproduce model runs
- + Are we ready to tackle these issues?
  - + Is anybody (beyond wishful thinking and lip-service) achieving results?
    - + ESMF?

Summary

Currently:

- + Multi−Dataset Visualization is
  - + Cool, Non−trivial effort, Non−automatable (without pre−processing)
  - + "Help me... help you!"
- + Assimilation is
  - + Much Harder
  - + Limited to pre−processed replicated files

Future: Grand Initiative Projects

- + Formidable technical hurdles include:
  - + Need for (unprecedented) cooperation among data creators / owners
    - + Webservers, etc.
  - + Powerful Data Model and supporting software libraries
    - + Unified (not Federated) to avoid combinatorial explosion of complexity
  - + Getting serious about *Fundamental Metadata*
    - + *Intrinsic Inequity Problem:* Data creators must generate metadata (burdensome) but gain no personal benefit for their efforts, while those that benefit (downstream re−users) incur no costs
  - + Paying attention to Efficiency / Performance for distributed HPC applications
- + And even more formidable management hurdles, especially stable funding!